

Preferences and Nonmonotonic Reasoning

Gerhard Brewka, Ilkka Niemelä,
and Mirosław Truszczyński

■ We give an overview of the multifaceted relationship between nonmonotonic logics and preferences. We discuss how the nonmonotonicity of reasoning itself is closely tied to preferences reasoners have on models of the world or, as we often say here, possible belief sets. Selecting extended logic programming with answer-set semantics as a generic nonmonotonic logic, we show how that logic defines preferred belief sets and how preferred belief sets allow us to represent and interpret normative statements. Conflicts among program rules (more generally, defaults) give rise to alternative preferred belief sets. We discuss how such conflicts can be resolved based on implicit specificity or on explicit rankings of defaults. Finally, we comment on formalisms that explicitly represent preferences on properties of belief sets. Such formalisms either build preference information directly into rules and modify the semantics of the logic appropriately or specify preferences on belief sets independently of the mechanism to define them.

One of the fundamental challenges of knowledge representation originates from a simple observation that information available to us more often than not is incomplete. Humans turn out to be quite effective at making decisions and taking actions when faced with incomplete knowledge. The key seems to be the skill of commonsense reasoning, based on our inherent preference to assume that things, given what we know, are normal or as expected. This assumption allows us to form preferred belief sets, base our reasoning exclusively upon them, and ignore all other belief sets that are consistent with our incomplete knowledge but represent situations that are abnormal or rare.¹ In this way, commonsense reasoning provides a handle on the complexity of the world around us by eliminating unlikely alternatives. The challenge for knowledge representation research has been to automate commonsense reasoning by finding formal ways to represent incomplete information and to specify preferred belief sets.

Even a simple example shows that the problem is real. Let us consider the task of representing different jobs at a university, relations among them, and their duties. For instance, being a professor is one type of a university job, and we want to model the fact that professors teach. Right away we face a difficulty. Professors teach, but there are exceptions to that rule. A more accurate statement is that professors *normally* teach. How to model such normative statements and how to reason about them is not at all obvious.

Classical logic, be it first-order or modal, is not the right solution. Given a base theory, a formal description of our knowledge as a set of formulas in the logic, there are no means to distinguish between its models. In other words, a base theory, if expressed and interpreted in classical logic, provides no information on which belief sets, the theories of models, might be preferred for reasoning—all must receive an equal consideration (see figure 1). This aspect of classical logic has several far-reaching consequences. First, there are no concise ways to represent lack of knowledge.

To represent the statement “professors normally teach” we might denote by p the property that somebody is a professor, and by t that she teaches. We might then consider using the implication

$$p \rightarrow t$$

to represent the statement in question.

If for some individual, we establish p (professor), then t (teaches) follows immediately. The problem is that the formula does not refer to normality; the representation becomes useless if the situation we deal with is abnormal (for instance, that particular individual is also a department chair).

Thus, we could introduce a notation ab to denote abnormal situations, and represent our rule by the implication

$$p \wedge \neg ab \rightarrow t.$$

The reading of this formula corresponds exactly to the statement we want to model. But now, to derive t , in addition to p (as before) we also need to establish $\neg ab$. To put it differently, given p and the formula $p \wedge \neg ab \rightarrow t$, there are three possible models (which we represent as sets of true propositions): $\{p, t\}$ and $\{p, ab\}$ and $\{p, ab, t\}$. Since classical logic does not prefer any, t cannot be derived. A departure from reasoning sanctioned by classical logic becomes necessary!

Figure 1. Professors Normally Teach Example.

Let us consider the theory

$$p \wedge \overset{p}{\neg ab} \rightarrow t$$

discussed previously. Circumscription with respect to the propositional variable ab prefers models that are *minimal* with respect to ab , and reasons with respect to preferred models only. For the “professors normally teach” example, just one model is preferred:

$$\{p, t\}.$$

Thus, under circumscription, t follows.

Figure 2. Professors Normally Teach Example: Circumscription.

Second, classical logic is *monotonic*: while new information may add to what we know, it does not sanction retractions of conclusions reached earlier. Or, to put it differently, new information may eliminate some models (belief sets) but never leads to new

ones. Yet the ability to retract or revise previous conclusions is one of the essential features of common-sense reasoning. Thus, a different breed of logics is needed to formalize it. Such logics, called *nonmonotonic*, started to emerge in the late 1970s and early 1980s (Reiter 1978, McCarthy 1980, Reiter 1980) and have been studied extensively ever since (Marek and Truszczyński 1993, Antoniou 1997).

The single common thread running through nonmonotonic logics is that they distinguish among belief sets and use only the preferred ones in reasoning. That is, they have precisely the feature that is missing from classical logic. Particular ways in which preferred belief sets are identified differ. They may be as simple as a restriction of models to minimal ones, as in circumscription (McCarthy 1980) (see figure 2) or, more generally, preference logic (Shoham 1987). They may be more involved, using a new syntax for describing rules premised on assumptions of consistency, called defaults, and a semantics designed to interpret defaults according to their intended meaning, as in default logic (Reiter 1980) or its simpler version, extended logic programming (Gelfond and Lifschitz 1990; 1991). Finally, in some cases, they exploit explicit user-provided preferences on defaults or on properties of belief sets, as in many variants of ranked default logic (Gelfond and Son 1998; Brewka and Eiter 2000), logic programming with ordered disjunction (Brewka 2002), or answer-set optimization (Brewka, Niemelä, and Truszczyński 2003).

When one thinks about preferred belief sets in the context of nonmonotonic reasoning, the term *preferred* often has a different meaning than that ascribed to it in decision theory. It refers to belief sets “preferred” by agents in reasoning because they give a most accurate picture of reality, and not to belief sets consisting of elements we would like to be true. To give an example, if we are normally tired, we prefer to reason under that assumption (we prefer realistic belief sets in which we are tired as the basis for our reasoning). But being tired is not our preference (in the decision-theoretic sense). Nonmonotonic logics are mainly concerned with identifying belief sets preferred for reasoning, assuming the world is as normal as possible. Nevertheless, some of the preference techniques of nonmonotonic systems, and even nonmonotonic logics by themselves, can be used to model both types of preference, and we will see examples of both types later in this article.²

In this article, we aim to give an overview of the multifaceted relationship between nonmonotonic logics and preferences. To make our discussion more concrete, we select one particular nonmonotonic logic, extended logic programming (Gelfond and Lifschitz 1990). The choice is not coincidental. Extended logic programming is a simple

fragment of default logic, yet it is rich enough to retain most of the essential features of the latter. It is one of the basic formalisms of answer-set programming, a popular paradigm for computational knowledge representation postulated in Niemelä (1999) and Marek and Truszczyński (1999). Moreover, extended logic programming is arguably the most successful nonmonotonic logic in terms of implemented systems.³ Finally, it has been extensively studied for possible extensions with explicitly specified preferences (Brewka and Eiter 1999; Schaub and Wang 2003).

We organize the article as follows. The next section provides a brief introduction to extended logic programming and discusses how the basic nonmonotonic semantics of answer sets of extended logic programs addresses the preference among belief sets and, in particular, how it represents and interprets normative statements. The following section is concerned with the use of implicit specificity of defaults to resolve conflicts among them and refine the notion of a preferred belief set. Conflicts among defaults can also be resolved by ranking them explicitly. This approach to the specification of preferred belief sets is covered next. Formalisms in which preference information is built into rules directly offer yet another way to specify preferred belief sets. We present one formalism that does just that, *logic programming with ordered disjunction* (Brewka 2002), whose semantics is based on the semantics of extended logic programs. For the first time, we encounter there a nonmonotonic reasoning system, which on the one hand defines belief sets preferred for reasoning (through the mechanism of ELP) and, in the same time, allows us to specify elements we prefer to have in them directly through ordered disjunctions. We follow with a discussion of formalisms that specify preferences on belief sets directly, independently of the mechanism to define them. A brief discussion and concluding remarks close the article.

The Basic Approach: Assuming Normality

Assuming normality leads to preferences on belief sets: those that describe normal or typical situations are preferred. As we noted in an earlier example, one way to model normality is through minimization of properties that are abnormal (see figure 2). In fact, one might argue that all major nonmonotonic logics and their semantics stem from different ways to deal with that issue. We illustrate this point using our nonmonotonic logic of choice: extended logic programming (ELP). Readers interested in a more detailed introduction to logic programming and answer set semantics may consult Gelfond and Lifschitz (1990), Gelfond and Leone (2002), or Baral (2003).

We assume a fixed set, say At , of propositional atoms. Atoms and their negations are *literals*.

A *rule* is an expression of the form

$$L_0 \leftarrow L_1, \dots, L_k \text{ not}(L_{k+1}), \dots, \text{not}(L_n)$$

where L_i are literals and **not** stands for the *default negation* (also referred to as *negation as failure to prove*).

The literal L_0 is called the *head* of the rule, the set $\{L_1, \dots, L_k \text{ not}(L_{k+1}), \dots, \text{not}(L_n)\}$ — the *body*. Literals L_1, \dots, L_k have positive and literals L_{k+1}, \dots, L_n have *negative* occurrences in the body of the rule.

Figure 3. ELP Rules.

Our discussion is restricted to the propositional case, which is sufficient for the illustration of all key ideas behind ELP. Figure 3 describes the syntax of rules, basic building components of programs. They involve the nonstandard negation connective *not* (in addition to the standard one, \neg).

A rule is meant to function as an inference rule. However, the intended interpretation of **not** is different from that of \neg . If **not**(L) appears as a premise in the body of a rule, to regard it as satisfied we must establish that L cannot be derived rather than derive $\neg L$. Thus, the intuitive reading of a rule is:

conclude the literal in the head if the positive literals in the body hold (have been derived) and no literal with a negative occurrence in the body can be derived.

For example, given a program consisting of the rules p and $t \leftarrow p, \text{not}(ab)$, we should conclude t as there is no way to derive ab , and so **not**(ab) holds by default (even though $\neg ab$ cannot be established). In this way, ELP can model the “professors normally teach” example. Another approach, not making a direct use of abnormality, is discussed in figure 4.

The intuition behind rules modeling normative statements can be made precise and can be extended to the case of general rules. First, we note that each program determines a class of sets of literals that are consistent with it. These sets of literals are formally defined as sets of literals closed under rules of the program and can be regarded as models of belief sets determined by the knowledge encapsulated by the program (figure 5).

Belief sets do not provide an intended interpretation of **not**, as the example in figure 5 demonstrates. Thus, the class of belief sets has to be narrowed down to a subclass of *preferred* belief sets or, more formally, *answer sets*, in order to capture the intended meaning of **not**. For some programs, defining answer sets is straightforward. If a program contains no occurrences of **not**, minimiza-

One way to express the statement that professors normally teach is to say that they teach if there is no explicit information that they do not. That latter statement can be translated literally into an ELP rule:

$$t \leftarrow p, \text{not}(\neg t).$$

If whenever an exceptional situation arises — a professor does not teach — there is information available to establish that, the rule seems indeed to be a correct representation of the normative statement it is supposed to model.

Figure 4. “Professors Normally Teach” in ELP.

A set U of literals is *closed under a rule* if:

1. U is inconsistent and contains all literals, or
2. U is consistent and contains the head of the rule whenever it contains all literals that occur positively and does not contain any literal that occurs negatively in the body of the rule.

Further, U is *closed under a program* if U is closed under every rule in the program. Such sets can be regarded as “consistent with the program.”

For the “professors normally teach” example, we have the following belief sets: $\{p, t\}$ and $\{p, \neg t\}$. The latter one does not capture the normality assumption and so, it is not preferred (there is no information that would imply $\neg t$ so the situation is assumed to be normal).

Figure 5. Belief Sets or Sets Closed under Rules.

tion provides the right concept. Indeed, every program without **not** has a unique least (and so minimal) belief set, and we accept this set as an answer set (the only one) of **not**-free programs (see figure 6).

We use the case of **not**-free programs as a springboard to the general case. Given an arbitrary program, we proceed as follows. First, we make an assumption concerning the status of the default literals $\text{not}(L)$ in the program by guessing the set of those literals that are implied by the program. If L is guessed to be derivable, $\text{not}(L)$ is assumed false; otherwise, $\text{not}(L)$ is assumed true. Second, we verify the guess. To this end, we simplify the program taking into account the logical values of the default literals. In this way, we obtain a program without **not**. If the answer set of this program (the

concept is well defined, as the program is “positive”) coincides with our guess, the guess is verified and is accepted as an *answer set* of the program that represents a preferred belief set (see figure 7).

In our discussion, we have stressed that nonmonotonic logics describe preferred belief sets, where the term *preferred* means preferred for reasoning. However, the basic mechanism of selecting preferred belief sets can also be used to model preferences that look more like typical decision-theoretic ones. For instance, if somebody prefers coffee to tea early in the morning, and tea to coffee at all other times, this preference can be represented by an ELP program consisting of the following two rules: $\text{tea} \leftarrow \text{not}(\text{early}), \text{coffee} \leftarrow \text{not}(\text{tea})$. This program has one answer set, $\{\text{tea}\}$, which reflects the preference for tea in a typical situation (here, typical means “not early”). However, if extended with the fact *early*, the resulting program has again just one answer set, $\{\text{coffee}\}$, correctly reflecting the corresponding preference for a drink.

Handling Conflicts Implicitly: Preferring More Specific Information

In the last section we have seen how the use of **not** in the body of ELP rules gives rise to the selection of preferred belief sets, called answer sets in the context of ELP. Other nonmonotonic formalisms have similar mechanisms for selecting preferred belief sets. For instance, Ray Reiter’s default logic (Reiter 1980), of which ELP is a special case, is based on a special kind of inference rules that are able to refer to the consistency of formulas. They are used to extend the set of beliefs of an agent. In circumscription (McCarthy 1980) various strategies for minimizing the extent of predicates can be specified. A standard approach here is to minimize abnormality predicates ab_i used to state that objects are abnormal in certain aspects. Similar mechanisms can be found in all nonmonotonic formalisms. In each case the goal is the same, namely to identify those belief sets in which everything is as normal as possible.

This basic approach still faces a difficulty: there are frequent cases where default rules support conflicting conclusions. One may thus end up with a large number of preferred belief sets that need to be narrowed down further.

In such a situation we can use preferences among rules to resolve conflicts. Two options are available: we can use preferences that are implicit in the available information, or we can use explicit preference information that needs to be specified by the knowledge engineer. We will deal with the first option in this section.

Sometimes there is a natural way to solve conflicts among rules. Consider again the example in

figure 7. Intuitively, the answer set in which the person does not teach ($\{c, p, \neg t\}$) is more plausible than the other one. This is because the conclusion $\neg t$ in this answer set depends on more specific information than the conflicting conclusion t in the other answer set: department chair is used to derive $\neg t$, professor to derive t , and since all department chairs are professors, it is natural to prefer information based on the former.

The notion of specificity—as an additional, implicit way of selecting preferred belief sets—has received considerable attention in the research community. In its purest form this notion was investigated in the context of nonmonotonic inheritance networks, a graphical formalism for representing simple defaults of the form $a \rightarrow b$ (normally, a 's are b 's) and $a \nrightarrow b$ (normally, a 's not are b 's). These defaults correspond to ELP rules $b \leftarrow a, \text{not}(\neg b)$ and $\neg b \leftarrow a, \text{not}(b)$, respectively. The main research goal is to capture adequately the notion of specificity and so, implicitly, that of a preferred belief set.

In our teaching example this looks like a rather simple task: *department chairs don't teach* is more specific than *professors teach* because department chairs are professors (and not vice versa). However, even in the context of simple inheritance networks coming up with an adequate formalization of specificity turns out to be more difficult than it may seem. What if, for instance, *department chairs are professors* is not a strict rule, but yet another default that may be defeated by more specific information? In fact, as the title of an early paper (Touretzky, Horty, and Thomason 1987) already indicates, for complex examples there exist different “clashing” intuitions leading to different formalizations, and it is somewhat difficult to tell which one is most appropriate in a particular case. For readers interested in more details we refer to John Horty's excellent overview article (Horty 1994).

An example of a rule-based system that implicitly prefers more specific information is system Z (Pearl 1990). It partitions a set of default rules R into sets R_0, R_1, \dots . A rule is in R_0 if adding its prerequisite to R does not lead to inconsistency. For the consistency check the rules are treated like ordinary implications. Similarly, a rule is in R_i if it is not in any of the sets R_{i-k} and adding its prerequisite does not render $R \setminus \{R_0, \dots, R_{i-1}\}$ inconsistent. Intuitively, this ranking gives higher priority to more specific rules. A similar ranking is used in Sarit Kraus, Daniel Lehmann, and Menachem Magidor's rational closure (Kraus, Lehmann, and Magidor 1990).

Handling Conflicts Explicitly: Ranking Rules

Specificity is not always sufficient to resolve conflicts. Conflicting rules are not necessarily in speci-

Department chairs (denoted by a propositional letter c) are professors. Deans (denoted by d) are not department chairs. If we also know d (an individual is a dean), we have the following program representing this knowledge:

$$\begin{array}{l} d \\ p \leftarrow c \\ \neg c \leftarrow d \end{array}$$

The set $\{d, \neg c\}$ is the only answer set of this program, being the least set of literals closed under it (the only other belief sets are: $\{d, \neg c, p\}$ and the set of all literals).

Figure 6. Not-Free Programs.

Let P be a program. Let us consider a set of literals U (a guess needed to determine the logical values of default literals). The *reduct* P^U of P with respect to U is obtained by removing from P all rules containing **not**(L) in the body, where $L \in U$, and by removing all default literals **not**(L) from the remaining rules. Clearly, the reduct P^U contains no occurrence of **not**.

We now define U to be an *answer set* of P if U is an answer set of P^U . One can show that answer sets of P are indeed belief sets of P . In fact, they are *inclusion-minimal* belief sets (the converse is not true in general).

To illustrate the construction, let us consider the following knowledge. Department chairs are professors, professors normally teach, department chairs normally do not teach. Let us also assume c (we are reasoning about a department chair). This knowledge can be formalized with the program P

$$\begin{array}{l} c \\ p \leftarrow c \\ t \leftarrow p, \text{not}(\neg t) \\ \neg t \leftarrow c, \text{not}(t) \end{array}$$

Consider now a set of literals $U_1 = \{c, p, t\}$ for which the reduct P^{U_1} consists of the first two rules in P and the rule $t \leftarrow p$. For P^{U_1} , U_1 is the inclusion-minimal belief set and, hence, an answer set of P . The program has also another answer set $U_2 = \{c, p, \neg t\}$ for which the reduct P^{U_2} consists of the first two rules in P and the rule $\neg t \leftarrow c$.

Figure 7. Answer Sets: The General Case.

ficity relation, and there may be other reasons to prefer one rule over another. For instance, rule preferences may be based on considerations of reliability, recency, or even authority, as in legal applications where rules representing federal law can

override rules representing state law. There may even be a good reason to prefer a less specific rule over a more specific one.

These additional, application-dependent preferences among rules can be handled by extending the underlying formalism and giving the knowledge engineer the possibility to specify preferences explicitly. A knowledge base (KB) now consists of a set of formulas or rules (depending on the formalism used) together with a preference relation on its elements. The preference relation can be an arbitrary ordering, but to keep the discussion simpler let us assume it is a total preorder (complete, transitive, reflexive). Such an ordering yields a partitioning of the KB elements into priority levels.

If the KB is, say, a set of propositional formulas, then it is not difficult to use the priority levels to construct preferred belief sets: just pick a maximal consistent subset of the most preferred formulas, add a maximal subset of the formulas in the next level consistent with the formulas chosen so far, and continue like this until all levels have been handled. This is exactly the preferred subtheory approach proposed in Brewka (1989). In the cardinality based proposal of Salem Benferhat and colleagues (Benferhat et al. 1993) also the *number* of elements chosen each time is taken into account; that is, a subset is added only if there is no consistent subset with more elements.

Applying these ideas to ELP, that is, defining the notion of a preferred answer set based on a prioritized logic program, is not as straightforward as it looks. The teaching example does not pose much of a problem. Assume we want to give lower preference to the information professors teach, that is, all rules in figure 7 except $t \leftarrow p$, $\text{not}(\neg t)$ are in the most preferred level, the latter rule is the single element in a (less preferred) second level. We can use the more preferred rules—which are not in conflict with each other—to compute the answer set in which the person does not teach ($\{c, p, \neg t\}$). Now the rule in the second level is no longer applicable, and the second, unintended, answer set is not generated.

In general, however, the following problem arises in rule-based approaches: the explicit preferences may be at odds with the implicit rule application ordering, which depends on the derivation of rule preconditions. Assume we have two conflicting rules r_1 and r_2 , and r_1 is preferred over r_2 . What if one of the preconditions of r_1 can only be derived using a rule r_3 that is even less preferred than r_2 ? Should r_1 still win in this case? In Brewka and Eiter (1999) a positive answer to this question is given, motivated by the view that r_1 should be applied if its missing precondition is accepted in the final answer set anyway. To define the preferred answer sets correspondingly, a second reduc-

tion, dual to the Gelfond-Lifschitz reduction (see figure 7), is used, which eliminates prerequisites of rules rather than **not**-literals. Other approaches have no preferred answer sets at all in such a situation. A detailed comparison of several approaches adding explicit preferences to ELPs can be found in Schaub and Wang (2001). A method for “compiling away” explicit preferences using additional control predicates in rules is described in Delgrande, Schaub, and Tompits (2003).

Building Preferences into Rules

In the last section, we discussed how to take into account preferences among rules. This approach still has some shortcomings. First, our preferences may depend on the current context: what is preferred under certain circumstances may not be preferred in other situations. Explicit preference orderings decide preference among rules once and for all and are not sufficiently flexible to take different contexts into account. Second, expressing preferences among rules can be viewed as an indirect way of expressing preferences on belief sets. Sometimes it is more convenient or necessary to express preferences directly in terms of the elements these belief sets contain. It is, in particular, true if we aim to represent conditional preferences differentiating between beliefs to be accepted based on other beliefs already accepted or rejected. *If I believe a, then among other beliefs I might adopt, I prefer b to c* is a simple example of a generic conditional preference. While we sometimes can handle such preferences by means of basic nonmonotonic logics (see our earlier tea-coffee example), in general, even extended with rankings on rules, nonmonotonic logics do not offer intuitive ways to model conditional preferences.

One way to deal with these limitations was proposed in Brewka, Niemelä and Syrjänen (2004). To express preferences among alternative beliefs, ordered disjunction (first proposed in Brewka, Benferhat, and Berre [2004]) is used in the heads of rules (the resulting programs are called *logic programs with ordered disjunction*, or LPODs). Ordered disjunctions are of the form $d = l_1 \times \dots \times l_n$ and say two things: (1) at least one of the literals l_i must be true, and (2) l_1 is the best alternative, l_2 is the second best, and so on. For example, the rule $\text{read} \times \text{walk} \leftarrow \text{rain}$ reads: if it rains, I prefer to read at home than to go out for a walk.

LPODs are designed to do two things at once. First, they determine the space of answer sets. To this end we associate with an LPOD P a family of extended logic programs and take their answer sets as answer sets of P . Second, LPODs use preferences encoded by ordered disjunctions in the heads of individual rules and aggregate them into a global preference relation on answer sets. Preferences

encoded by ordered disjunctions can be decision-theoretic (as in the aforementioned *read* versus *walk* example) or reasoning preferences (as in the example in figure 8).

We refer to Brewka, Niemelä, and Syrjänen (2004) for the details concerning the definition of the space of answer sets and explain only how LPODs determine global preferences on answer sets. We start by noting that satisfaction of an ordered disjunction rule r with $d = l_1 \times \dots \times l_n$ in the head is no longer a binary property. It becomes a matter of degree: if r 's body is satisfied and l_1 is in an answer set A then the satisfaction degree of r in A is 1, the best possible degree. If the body is satisfied, l_1 is not in A but l_2 is, then the degree is 2 and so on. Figure 8 provides an example.

In this manner, each rule ranks all answer sets based on its satisfaction degree. Since each rule provides its own ranking and we usually have more than one rule with ordered disjunction, we need a combination method that aggregates the individual rankings into a global preference order on the answer sets of the LPOD.

One can think of a variety of such combination methods. A rather cautious approach (in the sense that many optimal answer sets exist) is a Pareto-combination. Here an answer set A is at least as good as answer set A' just in case the satisfaction degree of each rule in A is at least as good (that is \leq) as its satisfaction degree in A' . Other methods compare the sets of rules satisfied to a particular degree in A and A' and prefer A if more rules have good satisfaction degrees, where *more* can be interpreted in terms of set inclusion or cardinality. Sometimes it is also useful to make numerical calculations on the satisfaction degrees for the comparison, for example, to just sum them up.

Decoupling Belief Set Construction and Ordering

In all formalisms discussed so far, including LPODs, the definition of the space of answer sets (belief sets) and preferences on them are intertwined complicating the process of preference elicitation. One approach to facilitating the elicitation process is to decouple the specification of the space of belief sets from the description of preferences on them. In addition to simplifying preference elicitation, that approach makes the overall setting better aligned with practical applications. Indeed, often what is possible is determined by external factors (available resources, constraints of the environment, design and engineering constraints for products) while what is preferred is described independently and by different agents (users, customers, groups of customers). In the decoupled approach typical applications are in product con-

We represent the information from figure 7. This time, we use *ab* literals. In addition, we represent the following knowledge: assuming department chairs are abnormal professors (with respect to teaching) is more reasonable than assuming they are abnormal department chairs. However, if we speak about German (*g*) department chairs, then it is more reasonable to assume they are abnormal department chairs (because in Germany department chairs actually have to teach):

$$\begin{array}{l} c \\ p \leftarrow c \\ t \leftarrow p, \text{not}(ab_1) \\ \neg t \leftarrow c, \text{not}(ab_2) \\ ab_1 \times ab_2 \leftarrow c, \text{not}(g) \\ ab_2 \times \bar{ab}_1 \leftarrow c, g. \end{array}$$

The program has two answer sets: $\{c, p, t, ab_2\}$ and $\{c, p, \neg t, ab_1\}$. The latter is preferred since it satisfies the fifth rule to degree 1, the former satisfies this rule to degree 2 only (the last rule is inapplicable and thus irrelevant for determining preferred answer sets). If we add *g*, the preferred answer set is $\{c, p, t, g, ab_2\}$ because the last rule is now applicable and better satisfied in this set than in $\{c, p, \neg t, g, ab_1\}$.

Figure 8. Programs with Ordered Disjunction.

figuration and constraint optimization and, therefore, instead of belief set (or answer set) a more commonly used term is *outcome* or *configuration*. We will adopt the first of these terms in this section.

Answer set optimization (ASO) programs introduced in Brewka, Niemelä, and Truszczyński (2003) are based on such a decoupling. Here the knowledge base is divided into two parts (P , $Prefs$) where P is a set of constraints (such as ELP rules) that define a collection of outcomes as answer sets of P and $Prefs$ is a set of preference rules of the form

$$C_1 > \dots > C_k \leftarrow a_1, \dots, a_n, \text{not}(b_1), \dots, \text{not}(b_m)$$

where the a_i s and b_j s are literals and the C_i s are Boolean combinations over atoms called ranking conditions. Such a rule intuitively reads: if an outcome S contains literals a_1, \dots, a_n and does not contain any of the literals b_1, \dots, b_m , then C_1 is preferred over C_2 , C_2 over C_3 , Hence, a rule ranks outcomes according to the satisfaction of the ranking conditions in its head. Given a collection of rules each outcome S can be assigned a satisfaction vector based on the satisfaction degrees of the rules in S and then outcomes can be ordered according to their satisfaction vectors. See figure 9 for more details. Figure 10 describes a configuration example illustrating the ideas.

The notion of optimal outcomes (see figure 9) is somewhat weak. In general, many optimal answer

Given a preference rule r and an outcome S , the *satisfaction degree* $v_s(r)$ of r in S is defined to be i if the body of r is satisfied by S , some ranking condition C_i is satisfied and i is the minimum index among the satisfied C_i s. Otherwise the degree is l (irrelevant). For example, consider an outcome $S = \{\text{ice-cream, beer, beef, soup}\}$ and a preference rule r

$$\text{red} \vee \text{beer} > \text{white} \leftarrow \text{beef}.$$

Now the satisfaction degree $v_s(r) = 1$ as the body and the first ranking condition of r are satisfied.

The satisfaction degrees are ordered in a preorder \geq where values l and 1 are equally good but better than all other degrees i for which $i \geq i + 1$ hold. For a set of preference rules $\{r_1, \dots, r_n\}$, an outcome S induces a *satisfaction vector* $(v_s(r_1), \dots, v_s(r_n))$ where $v_s(r_i)$ is the satisfaction degree of r_i in S . Outcomes are ordered as follows: $S_1 \geq S_2$ if $v_s(r_i) \geq v_{s_2}(r_i)$, for each $i \in \{1, \dots, n\}$ and $S_1 > S_2$ if $S_1 \geq S_2$ and for some $i \in \{1, \dots, n\}$, $v_{s_1}(r_i) > v_{s_2}(r_i)$. An outcome S is *optimal* if there is no outcome S' such that $S' > S$.

Figure 9. Ordering of Outcomes Based on Preference Rules.

sets may exist, and one often wants additional means to express that one preference rule is more important than another. This can be done by ranking preference rules such that rules with lower rank are preferred over rules with higher rank, see details in Brewka, Niemelä, and Truszczyński (2003).

Discussion

In this short overview we aimed to demonstrate the intertwined relationship between nonmonotonic logics and qualitative preferences.

First, nonmonotonic logics can themselves be considered as logics “with qualitative preferences,” allowing us to prefer some belief sets (theories of their models) to others. All major nonmonotonic logics, including preference logics, default logic, autoepistemic logic and extended logic programming, the formalism we discussed in more detail here, can be viewed in this way.

However, nonmonotonic logics by themselves are not flexible enough to represent a broad spectrum of preferences. Thus, additional principles, like that of specificity, are sometimes used to address the problem. In other cases, nonmonotonic logics allow for explicit extensions by rule (formula) orderings to expand their capabilities to represent preferred belief sets.

Nonmonotonic logics, whether with explicit

rule orderings or not, are not well designed for modeling orderings on belief sets specified in terms of preferences on their elements (or, to say it differently, using them to represent typical decision-theoretic preferences is often difficult or nonintuitive). In particular, they are not easy to use when conditional preferences are to be represented. Logic programs with ordered disjunction offer one way to address the problem.

Another, and more drastic, way is to use the basic mechanism of a nonmonotonic logic only to specify the space of possible belief sets (now often referred to as outcomes), and to describe preferences entirely outside of the logic in terms of preference rules. ASO programs illustrate that approach, bringing us to the end of our story.

However, we note that ASO programs are by no means the only game in town when preferences are specified separately from the space of outcomes. CP-nets (Boutilier et al. 1999) are another such formalism (see Francesca Rossi, Brent Venable, and Toby Walsh’s article “Preferences in Constraint Satisfaction and Optimization” in this issue for an introduction to CP-nets). In fact, the “decoupled” paradigm seems now to be a dominant approach to qualitative preference representation and reasoning. Thus, we conclude with comments pointing out the similarities and differences between ASO programs and CP-nets.

First, we note that CP-nets are concerned only with a fixed space of all configurations that contain for every variable exactly one value from its domain. In contrast, ASO programs offer substantial flexibility in defining the space of outcomes, through their use of extended logic programs as outcome-space generators. While hard constraints were proposed as a way to expand CP-nets with the capability of representing restrictions on outcomes, extended logic programming (and other formalisms based on nonmonotonic logics) seem to be superior from the knowledge representation point of view.

Next, unlike the original CP-nets, ASO preference rules allow us to specify *incomplete* conditional preferences on domain values of a variable (no information about preferences under some conditions) and *conflicting* conditional preferences on domain values of a variable (conflicting preferences under some conditions). These limitations of CP-nets have been addressed by recent extensions such as that in Wilson (2004). However, there is a key difference in the way the two approaches determine the preference ordering on the set of outcomes. CP-nets specify conditions that allow for direct comparisons of two outcomes by defining conditions under which we can transform one into another by means of an improving flip. Then, outcomes are compared by transitive closure of that relation. On the other hand, preference rules

assign a quality measure to each outcome (a satisfaction vector), and outcomes are ordered by comparing their quality measures by one of several methods developed in the literature.

Finally, we note that it is possible to integrate ideas behind ASO programs and CP-nets. An approach to extending ASO programs with CP-net type comparison of outcomes is developed in Brewka, Niemelä, and Truszczyński (2005).

Acknowledgments

We thank the reviewers for helpful comments. Ilkka Niemelä acknowledges the support of Academy of Finland grant 122399. Mirosław Truszczyński acknowledges the support of NSF grant IIS-0325063 and KSEF grant KSEF-1036-RDE.

Notes

1. By a belief set we mean a set of formulas describing a state of affairs the agent considers possible. In classical logic, belief sets correspond to the formulas true in a model. Nonmonotonic logics, such as default logic and autoepistemic logic, generate belief sets (called extensions, expansions, and the like) that are not necessarily complete. In any case, the fewer belief sets, the more certain is the agent about the state of affairs.

2. We thank Jérôme Lang for pointing out the need to emphasize the distinction and suggesting the example we used.

3. See, for instance, Gebser et al. (2007) for a report on the First Answer Set Programming System Competition in which 10 systems participated, as well as successful applications in planning, configuration, query answering, bioinformatics and space shuttle control (we refer to Leone et al. [2006] for a discussion of some of these applications, as addressed by the DLV system, one of the most advanced implementations of ELP).

References

- Antoniou, G. 1997. *Nonmonotonic Reasoning*. Cambridge, MA: The MIT Press.
- Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge, UK: Cambridge University Press.
- Benferhat, S.; Cayrol, C.; Dubois, D.; Lang, J.; and Prade, H. 1993. Inconsistency Management and Prioritized Syntax-Based Entailment. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI-1993*, 640–647. San Francisco: Morgan Kaufmann Publishers.
- Boutilier, C.; Brafman, R.; Hoos, H.; and Poole, D. 1999. Reasoning with Conditional Ceteris Paribus Preference Statements. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI-1999*, 71–80. San Francisco: Morgan Kaufmann Publishers.
- Brewka, G. 1989. Preferred Subtheories: An Extended Logical Framework for Default Reasoning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence, IJCAI-1989*, 1043–1048. San Francisco: Morgan Kaufmann Publishers.
- Brewka, G. 2002. Logic Programming with Ordered Disjunction. In *Proceedings of the 18th National Conference on*

Consider an example from Brewka, Niemelä, and Truszczyński (2003) where we are configuring a dinner and the possible outcomes (given using a set of rules P) include exactly one item from each of the following four sets:

$$\{\text{soup, salad}\}, \{\text{beef, fish}\}, \\ \{\text{pie, ice-cream}\}, \{\text{red, white, beer}\}.$$

Assume that the preference rules $Prefs$ are:

$$\text{white} > \text{red} > \text{beer} \leftarrow \text{fish} \\ \text{red} \vee \text{beer} > \text{white} \leftarrow \text{beef} \\ \text{pie} > \text{ice-cream} \leftarrow \text{beer}.$$

These rules designate as nonpreferred all outcomes containing *fish* but not *white*, all outcomes containing *beef* but not *red* or *beer*, and all outcomes containing *beer* and not *pie*. For example, consider the outcomes that contain *ice-cream* and *beer*:

$$S_1 = \{\text{ice-cream, beer, beef, soup}\} \\ S_2 = \{\text{ice-cream, beer, beef, salad}\} \\ S_3 = \{\text{ice-cream, beer, fish, soup}\} \\ S_4 = \{\text{ice-cream, beer, fish, salad}\}$$

Their satisfaction vectors are $V_1 = (l, 1, 2)$, $V_2 = (l, 1, 2)$, $V_3 = (3, l, 2)$, and $V_4 = (3, l, 2)$, respectively (see figure 9). Thus, S_1 and S_2 are equally good and are optimal among the four outcomes where S_3 and S_4 are equally good but less preferred than S_1 and S_2 .

Figure 10. Dinner Configuration Example.

Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence, AAAI/IAAI-2002, 100–105. Menlo Park, CA: AAAI Press.

Brewka, G., and Eiter, T. 1999. Preferred Answer Sets for Extended Logic Programs. *Artificial Intelligence* 109(1–2): 297–356.

Brewka, G., and Eiter, T. 2000. Prioritizing Default Logic. In *Intellectics and Computational Logic*, ed. S. Holldobler, 27–45. Dordrecht: Kluwer.

Brewka, G.; Benferhat, S.; and Berre, D. L. 2004. Qualitative Choice Logic. *Artificial Intelligence* 157(1–2): 203–237.

Brewka, G.; Niemelä, I.; and Syrjänen, T. 2004. Logic Programs with Ordered Disjunction. *Computational Intelligence* 20(2): 333–357.

Brewka, G.; Niemelä, I.; and Truszczyński, M. 2003. Answer Set Optimization. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-2003*, 867–872. San Francisco: Morgan Kaufmann Publishers.

Brewka, G.; Niemelä, I.; and Truszczyński, M. 2005. Prioritized Component Systems. In *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference, AAAI/IAAI-2005*, 596–601. Menlo Park, CA: AAAI Press.

Delgrande, J. P.; Schaub, T.; and Tompits, H. 2003. A Framework for Compiling Preferences in Logic Programs.

- Theory and Practice of Logic Programming* 3(2): 129–187.
- Gebser, M.; Liu, L.; Namasivayam, G.; Neumann, A.; Schaub, T.; and Truszczyński, M. 2007. The First Answer Set Programming System Competition. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning*, LPNMR-2007, Lecture Notes in Computer Science 4483, 3–17. Berlin: Springer.
- Gelfond, M., and Leone, N. 2002. Logic Programming and Knowledge Representation—The A-Prolog Perspective. *Artificial Intelligence* 138(1–2): 3–38.
- Gelfond, M., and Lifschitz, V. 1990. Logic Programs with Classical Negation. In *Proceedings of the 7th International Conference Logic Programming*, ICLP-1990, 579–597. Cambridge, MA: The MIT Press.
- Gelfond, M., and Lifschitz, V. 1991. *Classical Negation in Logic Programs and Disjunctive Databases*. New Generation Computing 9(3–4): 365–385.
- Gelfond, M., and Son, T. 1998. Reasoning with Prioritized Defaults. In *Proceedings of the Third International Workshop on Logic Programming and Knowledge Representation*, LPKR-1997, Lecture Notes in Computer Science 1471, 164–223. Berlin: Springer.
- Horty, J. F. 1994. Some Direct Theories of Nonmonotonic Inheritance. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, 111–187. New York: Oxford University Press.
- Kraus, S.; Lehmann, D. J.; and Magidor, M. 1990. Nonmonotonic Reasoning, Preferential Models, and Cumulative Logics. *Artificial Intelligence* 44(1–2): 167–207.
- Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Perri, S.; and Scarcello, F. 2006. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic* 7(3):499–562.
- Marek, V. W., and Truszczyński, M. 1993. *Nonmonotonic Logic; Context-Dependent Reasoning*. Berlin: Springer.
- Marek, V. W., and Truszczyński, M. 1999. Stable Models and an Alternative Logic Programming Paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*, ed. K. Apt, W. Marek, M. Truszczyński, and D. Warren, 375–398. Berlin: Springer.
- McCarthy, J. 1980. Circumscription—A Form of Nonmonotonic Reasoning. *Artificial Intelligence* 13(1–2): 27–39.
- Niemelä, I. 1999. Logic Programming with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence* 25(3–4): 241–273.
- Pearl, J. 1990. System Z: A Natural Ordering of Defaults with Tractable Applications to Nonmonotonic Reasoning. In *Proceedings of the Third Conference on Theoretical Aspects of Reasoning about Knowledge*, TARK-1990, 121–135. San Francisco: Morgan Kaufmann Publishers.
- Reiter, R. 1978. On Closed World Data Bases. In *Logic and Data Bases, Symposium on Logic and Data Bases*, Centre d'Études et de Recherches de Toulouse, 1977, ed. H. Gallaire and J. Minker, 55–76. New York: Plenum Press.
- Reiter, R. 1980. A Logic for Default Reasoning. *Artificial Intelligence* 13(12): 81–132.
- Schaub, T., and Wang, K. 2001. A Comparative Study of Logic Programs with Preference. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, IJCAI-2001, 597–602. San Francisco: Morgan Kaufmann Publishers.
- Schaub, T., and Wang, K. 2003. A Semantic Framework for Preference Handling in Answer Set Programming. *Theory and Practice of Logic Programming* 3(4–5): 569–607.
- Shoham, Y. 1987. A Semantical Approach to Nonmonotonic Logics. In *Proceedings of the Symposium on Logic in Computer Science*, LICS-1987, 275–279. Los Alamitos, CA: IEEE Computer Society.
- Touretzky, D. S.; Horty, J. F.; and Thomason, R. H. 1987. A Clash of Intuitions: The Current State of Nonmonotonic Multiple Inheritance Systems. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, IJCAI-1987, 476–482. San Francisco: Morgan Kaufmann Publishers.
- Wilson, N. 2004. Extending CP-nets with Stronger Conditional Preference Statements. In *Proceedings of the 19th National Conference on Artificial Intelligence and 16th Conference on Innovative Applications of Artificial Intelligence*, AAAI/IAAI-2004, 735–741. Menlo Park, CA: AAAI Press.

Gerhard Brewka is a professor of intelligent systems at University of Leipzig, Germany, where he also directs the doctoral program on knowledge representation. His main research interests are knowledge representation, in particular nonmonotonic reasoning, preference handling, inconsistency handling, reasoning about action, models of argumentation, and answer set programming. Brewka was program chair of ECAI-06, the European Conference on Artificial Intelligence, and KR-08, the International Conference on Principles of Knowledge Representation and Reasoning. In 2007 he received the AAAI outstanding senior PC member award. He is an ECCAI fellow and current president of ECCAI, the European Coordinating Committee on Artificial Intelligence.

Ilkka Niemelä is a professor of computer science at Helsinki University of Technology (TKK) where he is also chair of the degree program of computer science and engineering. His research interests include knowledge representation, answer set programming, automated reasoning, constraint programming, computer-aided verification, automated testing, and product configuration. Niemelä was program committee chair of LPNMR-2004, the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning, and ICLP-2007, the 23rd International Conference on Logic Programming. He is a member of the executive committee of the Association for Logic Programming.

Mirosław Truszczyński is a professor of computer science at the University of Kentucky. Truszczyński's research interests include knowledge representation, nonmonotonic reasoning, logic, logic programming, and constraint programming. He was a coauthor of the research monograph *Nonmonotonic Logic* that provided a comprehensive account of mathematical foundations of nonmonotonic reasoning, and a coeditor of *Logic Programming Paradigm: A 25-Year Perspective*, celebrating accomplishments of the field of logic programming. Truszczyński was program committee chair of LPNMR-2001, the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning, and ICLP-2006, the 21st International Conference on Logic Programming. He is a member of the advisory board of Knowledge Representation, Inc.